REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

AD-A273 134

2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
October 1993	Professional Paper

4. TITLE AND SUBTITLE

EVOLVING NEURAL NETWORK CONNECTIVITY

5. FUNDING NUMBERS

PR: ZW67
PE: 0601152N
WU: DN303002

6. AUTHOR(S)

J. R. McDonnell and D. Waagen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Command, Control and Ocean Surveillance Center (NCCOSC)
RDT&E Division
San Diego, CA 92152-5001

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
800 North Quincy Street
Arlington, VA 22217-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This work investigates the application of evolutionary programming, a stochastic search technique, for determining connectivity in feedforward neural networks. The method is capable of simultaneously evolving both the connection scheme and the network weights. The number of connections are incorporated into an objective function so that network parameter optimization is done with respect to network complexity as well as mean pattern error. Experimental results are shown for simple binary mapping problems.

Published in *Proceedings of the International Conference on Neural Networks*, 1993, Vol. II, pp. 863-868.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Neural Networks Evolutionary Programming Signal Detection	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAME AS REPORT

21a. NAME OF RESPONSIBLE INDIVIDUAL	21b. TELEPHONE (include Area Code)	21c. OFFICE SYMBOL
J. McDonnell	(619) 553-5762	Code 731

Evolving Neural Network Connectivity

John R. McDonnell & Don Waagen
NCCOSC, RDT&E Division
San Diego, CA 92152

Abstract-This work investigates the application of evolutionary programming, a stochastic search technique, for determining connectivity in feedforward neural networks. The method is capable of simultaneously evolving both the connection scheme and the network weights. The number of connections are incorporated into an objective function so that network parameter optimization is done with respect to network complexity as well as mean pattern error. Experimental results are shown for simple binary mapping problems.

I. INTRODUCTION

The neural network architecture design process is largely based on heuristics. Previous experience often dictates an initial network configuration for the problem at hand. If the network can be trained to achieve the designer's goals, the process is terminated. If success is not attained, a testing phase ensues which is largely trial and error. The result can often be a network with excess parameters and little regard for computational complexity.

In this research, a connectivity cost associated with the neural network configuration is incorporated into the optimization procedure in an effort to reduce the number of synapses and potentially even the number of nodes utilized in a network configuration. An optimized architecture offers increased throughput for near real-time signal processing applications as well as decreased memory requirements.

Simultaneously determining both network weight coefficients and structure requires a search procedure which is amenable to combinatorial optimization problems. The more successful algorithms for this these types of problems have generally been stochastic search techniques such as simulated annealing [1], genetic algorithms [2], and simulated evolution [3]. The simulated evolution, or evolutionary programming (EP), paradigm has been shown to have the desired attributes: combinatorial optimization capabilities [4], the ability to determine model structure [5], and the ability to train neural networks [6].

The premise of the current research is that near minimal size neural network architectures can be evolved using an objective function which includes a complexity term based on the neural network connectivity. Further, the proposed approach takes advantage of computational resources during the design/training phase thereby removing the burden of evaluation by trial-and-error from the designer. For purposes of discussion, Fig. 1 illustrates the structure of a hypothetically evolved neural network where the connectivity between neurons is determined via a multi-agent stochastic search technique. Nodes which are not connected can be pruned. This work extends previous research in evolving connectivity [7] and network architecture [8] by investigating a different mutation selection mechanism within the EP paradigm.

Similar work has been undertaken by Bornholdt and Graudenz [9] using an evolution strategy to determine both network structure and weight coefficients. Due to the generality of their implementation, recurrent networks can result requiring multiple sweeps to reach a stable state. The approach investigated in this work is limited to feed-forward networks. Other work has been done in removing connections during the training process. Hinton [10] allows weight elements to decay to zero unless reinforced otherwise. This is essentially the same as incorporating a term in the objective function which penalizes the use of non-zero weights [11]. Weight-elimination is proposed by Weigend *et al.* [12] which forces small weights to decay faster than large weights. Soft weight-sharing [13] also promotes magnitude reduction in small weights, albeit at greater complexity in the optimization process. Tenorio and Lee [14] have developed a self-organizing neural network (SONN) algorithm which constructs the network and determines the weights.

Weigend *et al.* [12] have demonstrated that the weight elimination technique outperforms "traditional nonlinear statistical approaches" in an effort to achieve minimally-sized networks for time-series prediction. Instead of a gradient search as employed in, [12], evolutionary programming is implemented as a global search technique. Baba [15] has demonstrated the effectiveness of using random optimization techniques over back propagation for training static networks. Genetic algorithms (GAs) have been also been used for constructing neural networks [16], although structure representation is sometimes unnecessarily complicated. Experiments conducted by Michalwicz [17] indicate that "the floating point representation is faster, more consistent from run to run, and provides a higher precision." Fogel [18] also demonstrates the higher precision capabilities of EP over GA from a convergence aspect.

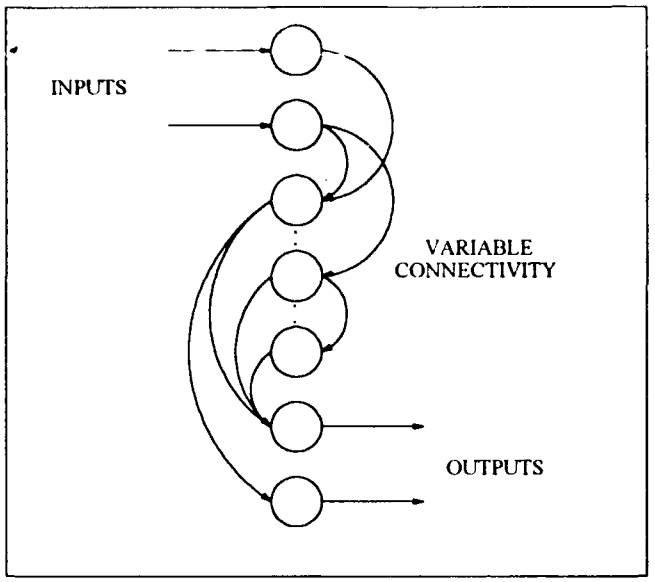

Fig. 1. A hypothetically evolved network structure with variable connectivity.

II. APPLYING EP TO NEURAL NETS

A. Evolutionary Programming

Evolutionary Programming is a neo-Darwinian search paradigm suggested by Fogel *et al.* [3]. This multi-agent search paradigm provides a systematic means for employing stochastic optimization techniques using an arbitrary objective function.

The EP optimization algorithm can be described by the following steps [5]:

1. Form an initial population $P_{2N-1}(x)$ of size 2N. The parameters x associated with parent element P_i are randomly initialized from a user specified search domain.
2. Assign a cost J_i to each element $P_i(x)$ in the population based on the objective function.
3. Reorder the population based on the number of wins generated from a stochastic competition process.
4. Generate offspring (P_N .... P_{2N-1}) from the highest ranked N elements (P_0 .... P_{N-1}) in the population by perturbing x.
5. Loop to step 2.

In addition to providing a systematic means of stochastic search, the generality of the EP optimization algorithm lends power to its implementation. The user is not bound to any particular coding structure nor mutation strategy. EP is used in this investigation since it is well suited for evolving both model structure and weight coefficients.

B. Determining Network Weights using EP

Evolutionary programming can be used for training static neural networks. The objective function can be the same as that used in back-propagation: minimize the sum-squared error function $E = 0.5\Sigma_p\Sigma_k(t_k - o_k)^2$ over all patterns p for k output neurons. The EP algorithm given in the previous section was employed to determine neural network weight coefficients.

A population of $2N$ feedforward networks is generated. Each network in the population is represented by a multidimensional weight array Φ_i with weights instantiated from a uniform $U(-0.5, 0.5)$ distribution. Next a cost is assigned to each network in the population. This cost is typically the mean sum-squared pattern error E given above. The "best" N members of the population generate offspring (perturbed weight sets) according to $W_o = W_p + \delta W_p$ where δW_p is a multivariate $N(0, S_f \cdot E_p)$ random variable with a scaling coefficient S_f and mean sum-squared pattern error E_p for each parent network. The scaling factor is a probabilistic analog to the stepsize used in gradient based methods and can even be treated as a random variable within the EP search strategy. The effect of the scaling factor is shown in Fig. 2 for sample training trials of the XOR mapping. The variance of the weight perturbations is bound by the total system error using this approach. To emulate the probabilistic nature of survival, a pairwise competition is held where individual elements compete against randomly chosen members of the population. For example, if network Φ_j is randomly selected to compete against network Φ_i, a "win" is awarded to network Φ_i if $J_i < J_j$. The N networks with the most wins are kept and the process is repeated.

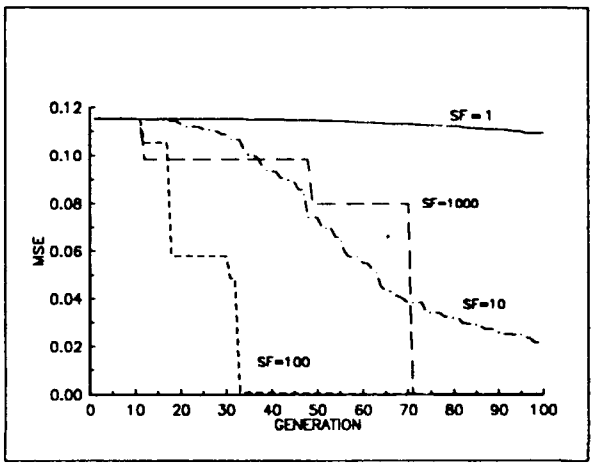

Fig. 2. EP training of a *2-2-1* XOR mapping network for various scaling factors and $N = 10$.

III. EVOLVING CONNECTIVITY

This section investigates neural network structural level adaptation as it pertains to the connectivity between neurons in a feedforward multi-layer architecture. The objective function has been modified to account for the *existence* of a connection. This is similar to the weight elimination technique [12] which minimizes an objective function of the form

$$J = E_p + \gamma \sum_{(ij)} \frac{(w_{ij} / w_o)^2}{1 + (w_{ij} / w_o)^2}$$

For $w_{ij} >> w_o$, the term inside the summation is nearly unity and the objective function can be approximated by

$$J \cong E_p + \gamma N_c$$

where N_c is the number of connections. When $w_{ij} << w_o$, the term inside summation approaches zero and has very little contribution to the magnitude of the objective function. If all of the input weights to a neuron become zero valued, the output of that neuron will serve as a bias within the network. The objective function proposed for this work is a linearly weighted combination of the number of connections N_c and the mean sum-squared pattern error

$$J = \alpha E_p + \beta N_c.$$

Instead of forcing the weight magnitudes to small values (which arbitrarily accentuates the search about the origin at the expense of outlying data points), the philosophy is taken that all connections are equally important. The network will benefit by simply disconnecting a synapse which has little effect on E_p. While the biological analog is that the synapse "dies", no such claims are made here. However, it is expected that the number of redundant hyperplanes will be reduced without introducing additional bias nodes which result from zero-input neurons. The number of active neurons can also be incorporated into the objective function to more fully account for the computational complexity. This is easily accommodated in the EP search procedure [7] by modifying the objective function given above so that

$$J = \alpha E_p + \beta N_c + \gamma N_n$$

where N_n is the number of active neurons in the network.

Analogous to the weight array, a connectivity array has been specified where $C_{lij} = 1$ if a connection exists or $C_{lij} = 0$ if no connection is present. A connectivity array that has all of its elements set to *1* yields a fully connected network. The designer must specify the number of hidden neurons over which the search is conducted. This number determines the maximum number of connections. In previous work, a state change was achieved by toggling randomly chosen connections [7] and a probabilistic approach was employed based on the variance of a neuron's activation level [8].

This work builds cumulative distribution functions (CDF's) based on the variance of a neuron's activation level to bias the connection selection process within the EP search. If additional connections are desired, a CDF is generated from all of the neurons in a layer which are not fully connected. The synapses propagating signals from very active neurons will tend to be chosen more than the synapses which propagate signals from relatively inactive neurons. The probability of connecting a synapse C_{lij} is given by

$$P_c(C_{lij}) = \frac{\sigma_{li}^2}{n_{li}^{(u)} \sum_{k \in l} \sigma_{lk}^2}$$

where $n^{(u)}_{li}$ is the number of unconnected synapses available for propagating the signal to the next layer and σ_{li} is the standard deviation of the activation level of neuron i in layer l over all input patterns for a fixed set of weights.

Likewise, if additional disconnections are desired, a CDF is generated from all of the connected neurons in a layer. The synapses propagating signals from neurons with relatively high activity levels will have a lower probability of being chosen to become disconnected. The probability of disconnecting a synapse C_{lij} is given by

$$P_d(C_{lij}) = \begin{cases} 1 & \text{if } \sigma_{li}^2 = 0 \\ \dfrac{\sigma_{li}^{-2}}{n_{li}^{(c)} \sum_{k \in l} \sigma_{lk}^{-2}} & \text{if } \sigma_{li}^2 \neq 0 \end{cases}$$

where $n^{(c)}_{li}$ is the number of synapses connected to the subsequent layer. Connections emanating from neurons with zero variance are always disconnected. Since the bias nodes are invariant, their connections remain present in the network.

To facilitate both structural level evolution as well as optimization of the weight coefficients, a strategy is employed that generates one set of offspring with a perturbed weight set in parallel with another set of offspring which has a modified connectivity structure. Fig. 3 illustrates the evolutionary search process over a single generation.

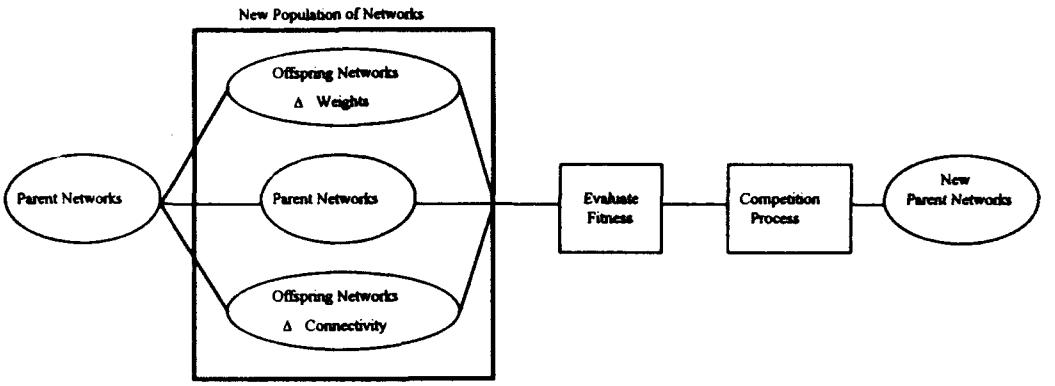

Figure 3. One generation of the evolutionary search process.

IV. RESULTS

A. The Parity Problem

The proposed approach for evolving network connectivity for the parity problem was considered. As a measure of sparseness of a network's connectivity, a dilution ratio has been defined [9] as $D = S/N^2$ where S is the total number of synapses and N is the number of neurons. Although the maximum number of neurons could be one of the search parameters [7], the available number of neurons were constant throughout these experiments. Starting from a fully connected 2-10-1 network, Fig. 4 shows the training results on the XOR mapping problem for S_f = 100, α=1 , β=0.001 and 10 parent networks. In this instance, the network evolved to a fully connected 2-2-1 network. Since 10 hidden units are available for connection, the dilution ratio for this network is D = 0.06 (not including connections to bias units which are not modifiable).

Fig. 5 shows a training session for the 3-bit parity problem. This session started from a fully-connected 3-10-1 network and S_f = 100, α=1 , β=0.0005 and 10 parent networks. The best (least cost) evolved network after 1000 generations has 13 connections as shown in Fig. 6. A fully-connected 3-3-1 network (12 connections) is able to achieve the mapping and would have resulted in lower cost solution. In the evolutionary process employed, only a single connection is modified each generation. This entails connecting a synapse to the third neuron in the hidden layer and disconnecting the synapse to the fourth neuron in the hidden layer- a two step process. As the search procedure arrives at a near optimal configuration, the probability of keeping a network over this two step process may be very small.

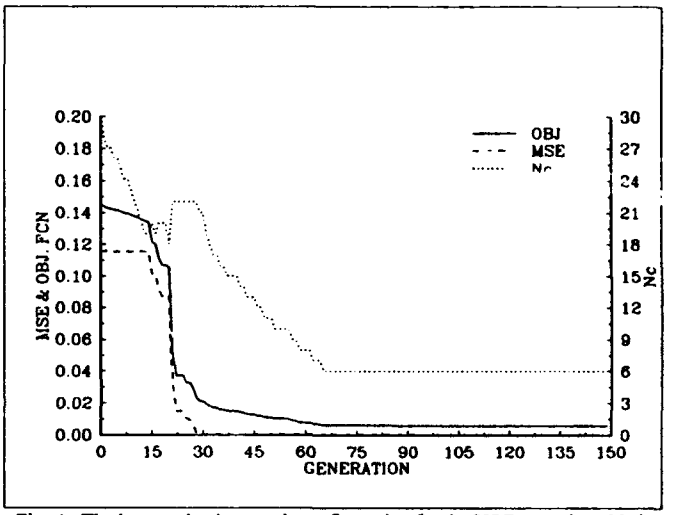

Fig. 4. The best evolved network configuration for the XOR mapping. Nc is the number of connections and OBJ is the objective function J.

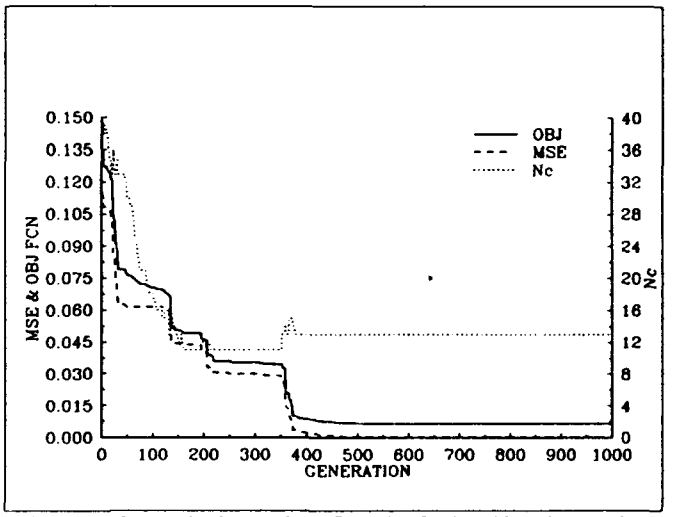

Fig. 5. The best evolved network configuration for the 3-bit parity mapping. Nc is the number of connections and OBJ is the objective function J.

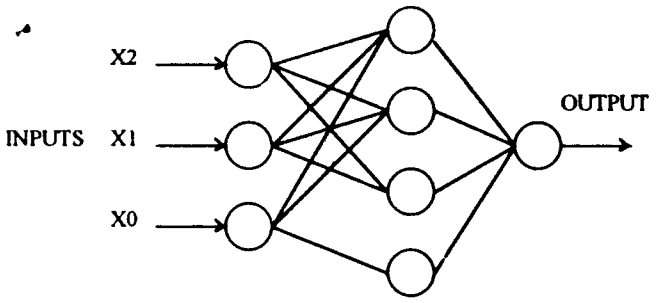

Fig. 6. The 3-bit parity mapping network resulting from the training session shown in Fig. 5. The bias connections are not shown.

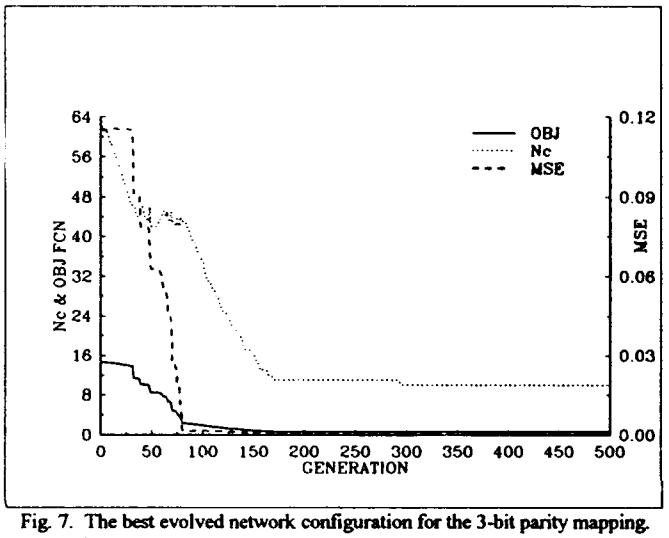

Fig. 7. The best evolved network configuration for the 3-bit parity mapping. Nc is the number of connections and OBJ is the objective function J.

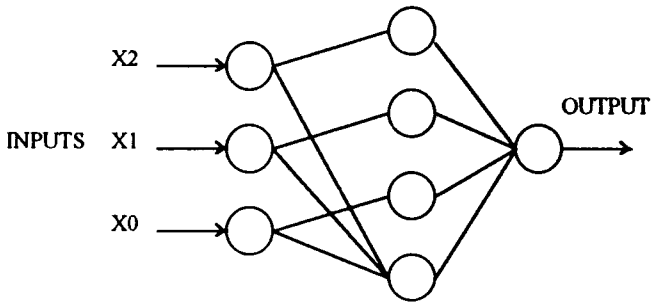

Fig. 8. The 3-bit parity mapping network which resulted from the training session shown in Fig. 7. The bias connections are not shown.

In the previous two examples the offspring weight sets were generated by perturbing the parent weight sets with a normally distributed $N(0, S_f \cdot J)$ random variable. Since this approach couples the perturbation magnitude to the networks objective function, an alternative technique was implemented were the weight sets were perturbed using a normally distributed $N(0, \alpha \cdot E_p)$ random variable. A sample training session using this approach on the 3-bit parity problem is shown in Fig. 7. This session started from a fully-connected 3-16-1 network with $S_f = 1$, α=100, β=0.05 and 10 parent networks. The resulting network has a diffusion ratio of D=0.039 as shown in Fig. 8 and achieved a MSE of 0.000018 after 2000 generations.

B. The T-C Problem

Additional experiments were done with a simple T-C pattern similar to that discussed by Rumelhart *et al.* [19]. This experiment involved distinguishing which 5 pixel 'T' or 'C' pattern exists on a 4x4 pixel grid. The training set consisted of 32 patterns which represented rotations in 90° increments and single pixel translations to the edges of the grid. Fig. 9 shows the training session starting from a fully-connect 16-16-1 network using the same parameters and weight perturbation approach given for the last example. The resulting network has a diffusion ratio of D=0.133 as shown in Fig. 10. The disconnected hidden units are also shown in Fig. 10 to illustrate that only those neurons which have inputs are propagating information forward in the network. The proposed approach can also be used to reduce the number of input connections. As shown in Fig. 10, pixel element [2,1] is not connected to the hidden units. An experiment was conducted using the XOR mapping augmented by an additional noisy input. The third (noisy) input consisted of a uniformly distributed $U(0,1)$ random variable. The evolved network had all of the synapses from the spurious input successfully disconnected.

V. CONCLUSION

The EP paradigm provides a robust framework for determining both model structure and weight coefficients. The enormity of the search space may hinder the simultaneous evolution of a good solution network from an individual parent network. A rigorous search is attempted by evolving disparate parameters with separate sets of offspring. This allows the optimization process to continue on the weight sets while various structures are being evaluated.

The biased EP search generally performed better than the random selection methods implemented in previous work [7,8]. This performance comes with the additional expense of building CDF's which is not required using a purely random selection process. The CDF's for layers without inputs are poorly defined (the sum of the variances of the activity levels over all of the neurons in the layer is zero). This situation may occur if an excess emphasis is placed on the number of connections in the objective function forcing N_c to zero. It is apparent that the proposed approach is very sensitive to the weighting parameters used in the objective function as well as the weight perturbation scaling factor.

The decoupled mutation strategy explored in this work may provide a basis for reducing this sensitivity. This is a topic for further investigation.

ACKNOWLEDGMENT

The authors thank Dr. A. Gordon, program director for NRaD internal research projects, for his support of this work.

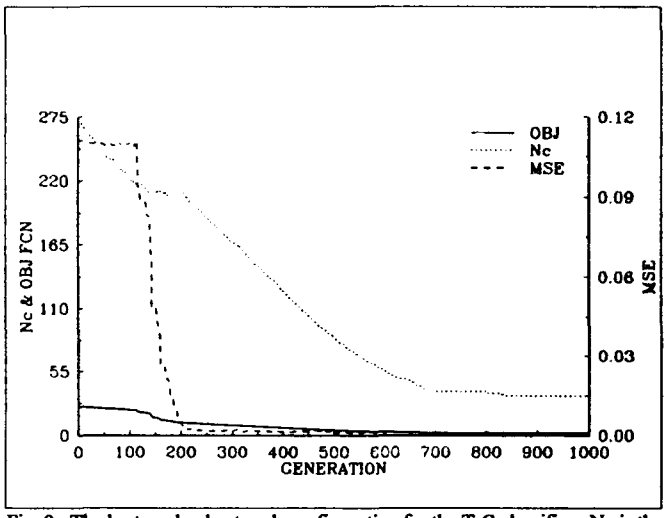

Fig. 9. The best evolved network configuration for the T-C classifier. Nc is the number of connections and OBJ is the objective function J.

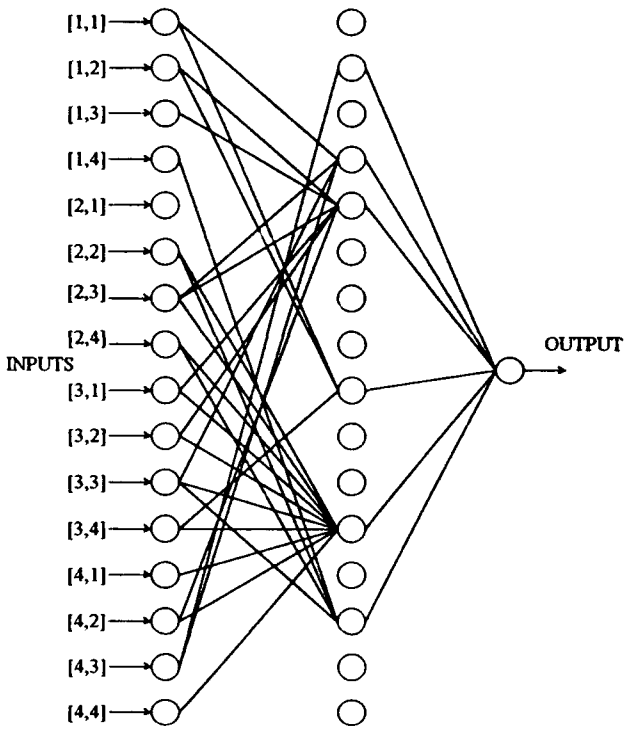

Fig. 10. The resulting network from the training session shown in Fig. 9. The bias connections are not shown.

REFERENCES

[1] E. Aarts and J. Korst, *Simulated Annealing and Boltzman Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*, John Wiley & Sons, 1989.

[2] J. H. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, 1992.

[3] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence through Simulated Evolution*, John Wiley & Sons, 1966.

[4] D. B. Fogel, "An evolutionary approach to the traveling salesman problem," *Biological Cybernetics*, Vol. 60, No. 2, 1988.

[5] D. B. Fogel, *System Identification though Simulated Evolution: A Machine Learning Approach to Modeling*, Ginn Press, Needham, MA, 1991.

[6] D.B. Fogel, L. J. Fogel, and V. W. Porto, "Evolving neural networks," *Biological Cybernetics*, Vol. 63, pp.487-493, 1990.

[7] J. R. McDonnell and D. Waagen, "Evolving neural network architecture," *SPIE Conf. on Neural and Stochastic Methods in Image and Signal Processing*, Vol. 1766, San Diego, 1992.

[8] J. R. McDonnell and D. Waagen, "Determining neural network connectivity using evolutionary programming", *Twenty-sixth Asilomar Conf. on Signals, Systems, and Computers*, Monterey, CA, 1992.

[9] S. Bornholdt and D. Graudenz, "General asymmetric neural networks and structure design by genetic algorithms", *Neural Networks*, Vol. 5, 1992.

[10] G.E. Hinton, "Connectionist learning procedures", Technical Report CMU-CS-87-115, Carnegie-Mellon University, Pittsburgh, PA, 1987.

[11] J. Hertz, A. Krogh, and R.G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, 1991.

[12] A. S. Weigend, D. E. Rumelhart, and B.A. Huberman, "Back-propagation, weight elimination and time series prediction", In *Proceedings of the 1990 Connectionist Models Summer School*, D. S. Touretzky (Ed.), Morgan Kaufmann, 1990.

[13] S.J. Nowlan and G.E. Hinton, "Simplifying neural networks by soft weight sharing", *Neural Computation*, Vol. 4, No. 4, 1992.

[14] M.F. Tenorio and W.T. Lee, "Self-organizing network for optimum supervised learning", *IEEE Trans. on Neural Networks*, Vol. 1, No. 1, March 1990.

[15] N. Baba, "A new approach for finding the global minimum of error function of neural networks", *Neural Networks*, Vol. 2, pp. 367-373, 1989.

[16] N. Dodd, "Optimisation of network structure using genetic techniques", *Int. Joint Conf. on Neural Networks*, San Diego, 1990.

[17] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, 1992.

[18] D.B. Fogel, "Asymptotic convergence properties of genetic algorithms and evolutionary programming: analysis and experiments", (unpublished).

[19] D.E. Rumelhart, G.E. Hinton and R.J. Williams, "Learning internal representations by error propagation", in *Parallel Distributed Processing, Volume 1*, D.E. Rumelhart and J.L. McClelland (Eds.), MIT Press, 1986.